AMENDMENTS TO THE CLAIMS

1. (Previously Presented) A method of creating run time executable code for a processing element array, comprising:

partitioning the processing element array into a plurality of hardware accelerators;

identifying a plurality of functions in a program source code that are anticipated to consume a substantial execution time;

decomposing the program source code into a plurality of kernel sections, wherein the identified plurality of functions are recognized as the plurality of kernel sections;

mapping said plurality of kernel sections into a plurality of hardware dependent executable code for execution on the plurality of hardware accelerators; and

forming a matrix describing different combinations of said plurality of hardware accelerators, code variants and said hardware dependent executable code to support run time execution of the plurality of kernel sections by the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.

2. (Original) The method of claim 1, wherein said partitioning includes partitioning into digital signal processors.

3. (Original) The method of claim 1, wherein said partitioning includes partitioning into bins.

4. (Currently amended) The method of claim 1, wherein said mapping includes mapping into multiple hardware contexts.

5. (Original) The method of claim 4, wherein said mapping into multiple hardware contexts includes mapping a first set of variants.

6. (Original) The method of claim 5, wherein said first set of variants are produced based upon resource usage.

7. (Currently amended) The method of claim 5, wherein said mapping includes mapping a second set of variants ~~of said designs~~ configured to support multiple hardware configurations of one of a plurality of bins.

8. (Original) The method of claim 1, wherein said mapping is performed by a place and route.

9. (Original) The method of claim 1, wherein said decomposing is performed manually.

10. (Original) The method of claim 1, wherein said decomposing is performed by a software profiler.

11. (Currently amended) The method of claim 10, wherein said decomposing includes executing code compiled from said program ~~description~~ source code and monitoring timing of said executing.

12. (Original) The method of claim 11, wherein said executing utilizes a set of test data.

13. (Original) The method of claim 11, wherein said monitoring includes determining functions that consume a significant portion of said timing of said executing.

14. (Previously Presented) The method of claim 1, wherein said identifying includes identifying functions by identifying regular structures.

15. (Previously Presented) The method of claim 1, wherein said identifying includes identifying functions with a limited number of inputs and outputs.

16. (Previously Presented) The method of claim 1, wherein said identifying includes identifying functions with a limited number of branches.

17. (Currently amended) The method of claim ~~10~~ 1, wherein decomposing identifies overhead sections.

18. (Original) The method of claim 1, wherein mapping includes creating microcode.

19. (Original) The method of claim 1, wherein said mapping includes creating context dependent configurations.

20. (Original) The method of claim 1, wherein said matrix is sparsely-populated.

21. (Original) The method of claim 1, wherein said matrix is fully populated.

22. (Currently amended) A system for creating run time executable code for execution on a processing element array, comprising:

a plurality of hardware accelerators partitioned from the processing element array;

a plurality of kernel sections identified as ~~the~~ functions that are anticipated to consume a substantial execution time ~~in~~ of a program source code~~, for~~ when ~~execution~~ executed on said plurality of hardware accelerators;

a plurality of hardware dependent executable code derived from said kernel sections for execution on said plurality of hardware accelerators; and

a matrix describing different combinations of said hardware accelerators, code variants and said hardware dependent executable code configured to support run time execution on the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.

23. (Currently amended) The system of claim 22, wherein said hardware accelerators ~~includes~~ include digital signal processors.

24. (Currently amended) The system of claim 22, wherein said hardware accelerators ~~includes~~ include bins.

25. (Original) The system of claim 24, wherein said bins support multiple hardware contexts.

26. (Original) The system of claim 25, wherein said bins support a first set of variants configured to support said multiple hardware contexts.

27. (Original) The system of claim 26, wherein said first set of variants are produced based upon resource usage.

28. (Currently amended) The system of claim 27, wherein a second set of variants ~~of said designs are~~ is configured to support multiple hardware configurations of one of said ~~plurality of~~ bins.

29. (Original) The system of claim 22, wherein said mapping is performed by a place and route.

30. (Original) The system of claim 22, wherein said decomposing is performed manually.

31. (Original) The system of claim 22, wherein said decomposing is performed by a software profiler.

32. (Currently amended) The system of claim 31, wherein said software profiler executes code compiled from said program ~~description~~ source code, and monitors time consumed.

33. (Original) The system of claim 32, wherein said software profiler includes a set of test data.

34. (Original) The system of claim 32, wherein said software profiler determines functions that consume a significant portion of said time consumed.

35. (Original) The system of claim 31, wherein said software profiler is configured to identify kernel sections by identifying regular structures.

36. (Original) The system of claim 31, wherein said software profiler is configured to identify kernel sections by identifying sections with a limited number of inputs and outputs.

37. (Original) The system of claim 31, wherein said software profiler is configured to identify kernel sections by identifying sections with a limited number of branches.

38. (Original) The system of claim 31, wherein said profiler identifies overhead sections.

39. (Currently amended) The system of claim 22, wherein said ~~designs include~~ hardware dependent executable code includes microcode.

40. (Original) The system of claim 39, wherein said microcode includes context dependent configurations.

41. (Original) The system of claim 22, wherein said matrix is sparsely-populated.

42. (Original) The system of claim 22, wherein said matrix is fully populated.

43. (Previously Presented) A machine-readable medium having stored thereon instructions for execution by a processing element array, which when executed by said processing element array perform the following:

identifying a plurality of functions in a program source code that are anticipated to consume a substantial execution time;

decomposing the program source code into a plurality of kernel sections, wherein the identified plurality of functions are recognized as the plurality of kernel sections;

mapping said plurality of kernel sections into a plurality of hardware dependent executable code of execution on the plurality of hardware accelerators; and

forming a matrix describing different combinations of said plurality of hardware accelerators, code variants and said hardware dependent executable code to support run time execution of the plurality of kernel sections by the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.

44. (Previously Presented) A system configured to create run time executable code for execution by a processing element array, comprising:

means for identifying a plurality of functions in a program source code that are anticipated to consume a substantial execution time;

means for decomposing the program source code into a plurality of kernel sections, wherein the identified plurality of functions are recognized as the plurality of kernel sections;

means for mapping said plurality of kernel sections into a plurality of hardware dependent executable code for execution on the plurality of hardware accelerators; and

means for forming a matrix describing different combinations of said plurality of hardware accelerators, code variants and said hardware dependent executable code to support run time execution of the plurality of kernel sections by the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.